Requirements Engineering Aspects of a Geographically Distributed Architecture

Maria Spichkova and Heinz Schmidt
RMIT University, Melbourne, Australia
{*maria.spichkova, heinz.schmidt*}@*rmit.edu.au*

Keywords: Requirements Engineering, Geographically Distributed Development, Software and Systems Engineering

Abstract: We present our ongoing work on requirements specification and analysis for the geographically distributed software and systems. Developing software and systems within/for different countries or states or even within/for different organisations means that the requirements to them can differ in each particular case. These aspects naturally impact on the software architecture and on the development process as a whole. The challenge is to deal with this diversity in a systematic way, avoiding contradictions and non-compliance. In this paper, we present a formal framework for the analysis of the requirements diversity, which comes from the differences in the regulations, laws and cultural aspects for different countries or organisations. The framework also provides the corresponding architectural view and the methods for requirements structuring and optimisation.

1 INTRODUCTION

Globalisation of the software and systems development offer great opportunities for the development industry. However, they also mean new challenges coming from the diversity of requirements on different locations, especially in the sense of legal requirements and regulatory compliance. The reasons for the differences in software and systems requirements within different countries, states and organisations are the cultural and economics diversity, as well as the diversity in standards, legal regulations and laws. These challenges have some similarities with the problems of product customisation and the development of product lines, but the core of the architecture requirements is different due the specific nature of the relations between the requirements for the geographically and organisationally distributed development and application of software.

Requirements engineering (RE), i.e., requirements elicitation, evaluation, specification, and design producing the functional and non-functional requirements, is one of the key disciplines in the software development domain. It has a critical impact on the product's quality. Requirements-related errors are often a major cause of the delays in the product delivery and development costs overruns, cf. e.g., (van Lamsweerde, 2008; Pretschner et al., 2007; Rinke and Weyer, 2007). There are several methodologies on development of software systems from requirements, also enclosing CASE tools support, e.g., (Broy and Slotosch, 2001; Hölzl et al., 2010; Spichkova, 2011; Spichkova et al., 2012). The RE task is challenging even in the case of a local (non-distributed) development of a product for application within/for a single country or organisation, i.e. where the system acts within an environment with a uniform set of standards, legal regulations, etc. Thus, in the case of global development we need to have an approach that deals the corresponding issues in a systematic and scalable way. There are several approaches that check requirements for compliance (Breaux et al., 2008; Maxwell and Anton, 2009; Siena et al., 2009b) or ensuring the compliance of the outcomes of business processes against outcome-focused regulations (Yin et al., 2013). We are going a step further, aiming to cover the RE aspects for the geographically distributed development and application. To minimise the overall effort while specifying and also ensuring required software and system requirements in a global development context, we elaborated a logical framework. The framework provides methodological structuring the requirements for the geographically distributed product development and application, as well as developing of the corresponding global architecture requirements. The purposed approach will help to analyse the relations between requirements and to trace requirements' changes in a global context.

Outline: The rest of the paper is structured as follows: Section 2 overviews related work on integration architecture and RE as well as on regulatory compliance within the field of software engineering. In Section 3 we present our view on the architectural dependencies for the requirements for a global development and geographically distributed product application. In Section 4 we introduce the core ideas of requirements structuring process and requirements analysis in a global context. Section 5 concludes the paper by highlighting the main contributions of the paper and describing the future work directions.

2 RELATED WORK

Glinz (2007) surveyed the existing definitions of non-functional requirements (NFR), highlights and discusses the problems with the current definitions, and contributes concepts for overcoming these problems. In our work, we are mainly focusing on NFR in the sense of legal aspects and regulatory compliance, also taking into account human factor aspects of the requirement modelling (Spichkova, 2012).

Integrating Architecture and Requirements Engineering: The main purpose of the requirements specification (RS) is to elicit and to document the given problem (product/software/system) using concepts from the problem domain, i.e. on the RE phase we are speaking only on the *problem statement*. In contrast to this, the aim of a software architecture (SA) is to design a *draft of the solution* for the problem described in the RS, at a high level of abstraction. Thus, there are tight interdependencies between functional/non-functional requirements and architectural elements, which makes the integration of the RE and architecture crucial (In et al., 2001; Egyed et al., 2001). The results of the empirical study conducted by Ferrari and Madhavji (2007) also have shown that the software architects with the knowledge and experience on RE perform better, in terms of architectural quality, than those without these knowledge end experience.

Nuseibeh (2001) described a spiral model-like development cycle of requirements and architecture. Pohl and Sikora (2007) went further and have provided methodical guidance for the co-design. An experience-based approach for integration architecture and RE is presented by Paech et al. (2003). This approach that supports the elicitation, specification and design activity by providing *experience* in terms of questionnaires, checklists, architectural patterns and rationale that have been collected in earlier successful projects and that are presented to developers to support them in their task.

The REMseS approach (Braun et al., 2014) aims at supporting RE processes for software-intensive embedded systems. The authors introduced fundamental principles of the approach and gave a structural overview over the guide and the tool support.

In contrast to these approaches, our research covers the regulatory compliance aspects and is oriented on legal requirements representation and analysis in the scope of the global software development.

Regulatory compliance: A survey of efforts to support the analysis of legal texts in the context of software engineering is presented in (Otto and Anton, 2007). The authors discuss the role of law in requirements and identify several key elements for any system to support the analysis of regulatory texts for requirements specification, system design, and compliance monitoring. Nekvi et al. (2011) also identified key artefacts, relationships and challenges in the compliance demonstration of the systems requirements against engineering standards and government regulations, also providing This work provides a basis for developing compliance meta-model.

Kiyavitskaya et al. (2008) investigated the problem of designing regulation-compliant systems and, in particular, the challenges in eliciting and managing legal requirements. Breaux et al. (2006) reported on an industry case study in which product requirements were specified to comply with the U.S. federal laws. Maxwell and Anton (2009) performed a case study using our approach to evaluate the iTrust Medical Records System requirements for compliance with the U.S. Health Insurance Portability and Accountability Act. Siena et al. (2009a) presented the guiding rules and a framework for for deriving compliant-by-construction requirements, also focusing on the U.S. federal laws.

In contrast to these approaches, our research is oriented on global software development, dealing with diversity in standards, legal regulations, etc. within different countries and organisations.

3 GEOGRAPHICALLY DISTRIBUTED ARCHITECTURE

Suppose that we have to develop M products (software components/systems) $P_1, \dots, P_M$. We denote the set of products by P. Each product P_i, $1 \leq i \leq M$ has the corresponding set of requirements R_{P_i}. However,

in the case the legal requirements and the regulatory compliance are taken into account, the set of requirements will depend on the regulations and laws of the country/state the product is developed for.

In the case of global and remote development (Spichkova et al., 2013), we have to deal with cultural and economics diversity, which also has an influence on the software and system requirements. Suppose the products $P_1, \dots, P_M$ are developed for application in N countries $C_1, \dots, C_N$ with the corresponding

- regulations/laws $RegulC_1, \dots, RegulC_N$ and
- cultural/economics, i.e., human factor (Borchers, 2003), influences $HFC_1, \dots, HFC_N$.

We denote the set of requirements to the product P_i valid for the country C_j by $R_{P_i}^{C_j}$. The complete set of requirements to the product P_i is then defined by

$$R^{P_i} = \bigcup_{j=1}^{N} R_{P_i}^{C_j} \quad (1)$$

The sets of requirements R_{P_i} might be different for each product P_i in different countries, i.e. $R_{P_i}^{Cj1}$ is not necessary equal to $R_{P_i}^{Cj2}$ for the case $j1 \neq j2$.

Figure 1 presents the corresponding architectural dependencies for the requirements based on the regulations and laws of the country C_j. We divide the set of requirements R_{P_i} in two (disjoint) subsets. For each of these subsets, we have to distinguish two separate parts: *general* and *country-specific*:

- $RL_{P_i}^{C_j}$ denotes the requirements based or depending on the regulations and laws, which could be country/state/organisation-specific. Requirements of this kind does not depend on the human factor related aspects.

$$RL_{P_i}^{C_j} = RLgeneral_{P_i}^{C_j} \cup RLspecific_{P_i}^{C_j} \quad (2)$$

$$RLgeneral_{P_i}^{C_1} = \dots = RLgeneral_{P_i}^{C_N} \quad (3)$$

- $RFN_{P_i}^{C_j}$ denotes the functional and non-functional requirements that are independent from the regulations and laws, but may depend on the human factor related aspects, which could be country-specific.

$$RFN_{P_i}^{C_j} = RFNgeneral_{P_i}^{C_j} \cup RFNspecific_{P_i}^{C_j} \quad (4)$$

$$RFNgeneral_{P_i}^{C_1} = \dots = RFNgeneral_{P_i}^{C_N} \quad (5)$$

For simplicity, we denote the general subsets by $RLgeneral_{P_i}$ and $RFNgeneral_{P_i}$ respectively.

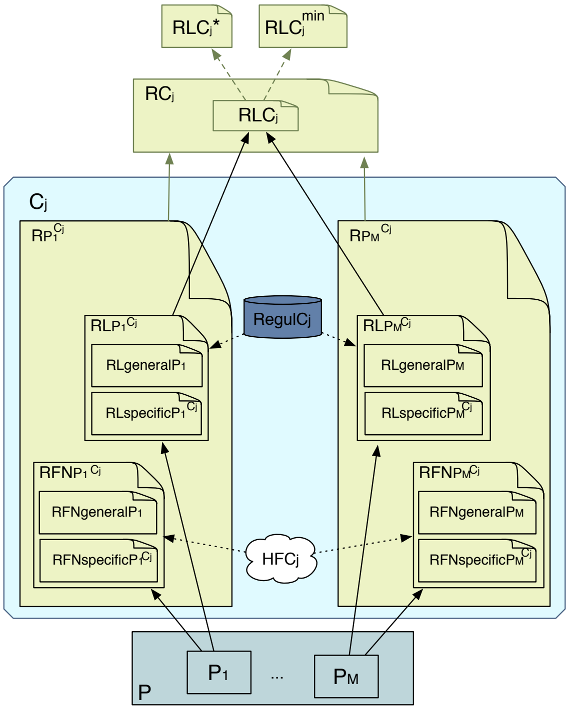

Figure 1: Architectural dependencies for the requirements based on the regulations and laws of the country C_j

Given the set $RL_{ALL}^{C_j}$ be the set of all legal requirements of the country C_j, we can see the set $RL_{P_i}^{C_j}$ as a *projection* of $RL_{ALL}^{C_j}$ to the product P_i:

$$RL_{P_i}^{C_j} = RL_{ALL}^{C_j}|_{P_i} \quad (6)$$

On this basis, we can say that the set of legal requirements for the country C_j related to the development of products $P_1, \dots, P_M$ is defined as a union over the corresponding requirements sets:

$$RL^{C_j} = \bigcup_{i=1}^{M} RL_{P_i}^{C_j} \quad (7)$$

However, it is not efficient to base the requirements analysis for the geographically distributed development on the sets RL^{C_j}:

- The sets $RegulC_1, \dots, RegulC_N$ could have a joint subset $Regul$ of the regulations/laws that are equal for all countries $C_1, \dots, C_N$:

$$Regul = RegulC_1 \cap \dots \cap RegulC_N \quad (8)$$

We denote for each $RegulC_j$ the corresponding compliment to $Regul$ (i.e. the country-specific subset) by $RegulC_1' = RegulC_1 \setminus Regul$.
It it is important to identify these subsets as well as the corresponding subsets of $RL^{C_1}, \dots, RL^{C_N}$, as this helps to trace the RL-requirements' changes more efficiently.

- Some requirements in RL^{C_j} can be stronger versions of another requirements from this set. For example, if $req_1 \in RL^{C_{i1}}$ and $req_2 \in RL^{C_{i2}}$ are not equal, they both will belong to RL^{C_j}, even if req_1 is a refinement of req_2. In this case, we call req_2 a *weaker version* of req_1 and denote it by $req_1 \rightsquigarrow req_2$.

Thus we need to have a structured architecture for the legal requirements on the products $P_1, \dots, P_M$. For this reason we build the corresponding sets $RL^{C_j\,min}$ and $RL^{C_j\,*}$, where $RL^{C_j\,*}$ denotes the strongest set of legal requirements for the country C_j (related to the concrete products development), and $RL^{min}_{C_j}$ denotes the set of legal requirements that should be fulfilled by *all* the products $P_1, \dots, P_M$ developed in the country C_j:

$$RL^{C_j\,min} = RL^{C_j}_{P_1} \cap \dots \cap RL^{C_j}_{P_M} \tag{9}$$

$RL^{C_j\,*}$ is an optimisation of RL^{C_j}, where all the weaker versions of the requirements are removed using the algorithm presented in Section 4.

4 REQUIREMENTS STRUCTURING AND ANALYSIS

In our framework, we perform the analysis based on the optimised views on the requirements sets, focusing on the regulatory/legal aspects. First of all, we analyse the sets of relevant regulations $RegulC_1, \dots, RegulC_N$. Three cases are possible:

- In the case $Regul = \emptyset$, we have the situation when the regulations are completely different for all $C_1, \dots, C_N$. This also implies that $RLgeneral_{P_i} = \emptyset$ for all P_i, $1 \leq i \leq M$, i.e., $RL^{C_j}_{P_i} = RLspecific^{C_j}_{P_i}$. We have to trace all the sets $RLspecific^{C_j}_{P_i}$ separately: changes in $RegulC_j$ do not influence on the global development process.
- The regulations are not completely identical for $C_1, \dots, C_N$, but $Regul \neq \emptyset$. If we can rely on the statical nature of this requirements (i.e. that these requirements do not change over the time of the development and the application), it would be beneficial to apply the component-based development paradigm (Crnković et al., 2007): the requirements $RLgeneral_{P_i}$ or at least the major part of them should correspond to architectural components(s) that are separate from the components corresponding to $RLspecific^{C_1}_{P_i}, \dots, RLspecific^{C_N}_{P_i}$. However, if any of regulations sets $RegulC_j$ has some changes, this would influence on the development process as a whole.
- In the case $Regul = RegulC_1 = \dots = RegulC_N$, we have the situation when the regulations are completely identical for all $C_1, \dots, C_N$, which also means $RL^{C_j}_{P_i} = RLgeneral_{P_i}$. If we can rely on the statical nature of this requirements, we have the simplest case for the development process: we develop a single component (system) from $RLgeneral_{P_i}$ to apply if for all $C_1, \dots, C_N$.

Thus, $RL^{C_j\,min}$ is defined on basis of $Regul$. The corresponding product-centred view on the architectural dependencies is presented on Figure 2. The algorithm of constructing $R^{C_j\,min}$ is trivial: we check all the requirements in $R^{C_j}_{P_1}, \dots, R^{C_j}_{P_M}$ to find out those elements, which belong to each of the sets.

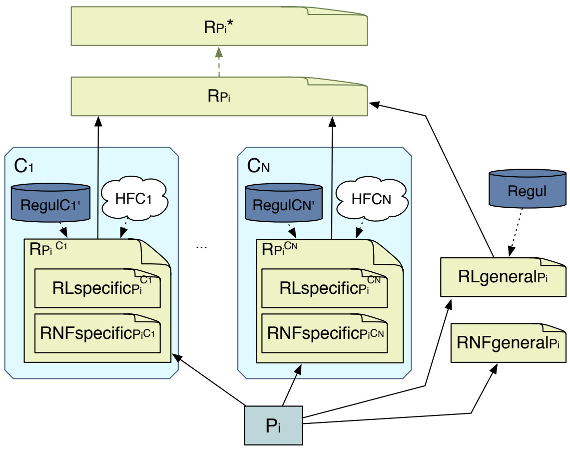

Figure 2: Architectural dependencies for requirements specified for the product P_i

Our approach is based on the ideas of refinement-based specification and verification. On the formal level, we need to define which exactly kind of a refinement we mean, e.g., behavioural, interface or conditional refinement (Spichkova, 2008; Broy and Stølen, 2001). However, on the level of the logical architecture and modelling of the dependencies between the (sets of) requirements, we can abstract from these details.

In this paper, we present a simplified version of the optimisation algorithm. It can be applied to build the set $RL^{C_j\,*}$ on the basis of RL^{C_j} (cf. Figure 1 for the country-centred view), to build the set $R_{P_i}{}^*$ (cf. Figure 2 for product-centred view) as well as to build the strongest global set of requirements R^* over $C_1, \dots, C_N$ (cf. Figure 3). We start the algorithm with an empty set and build it up iteratively from the elements of the corresponding set.

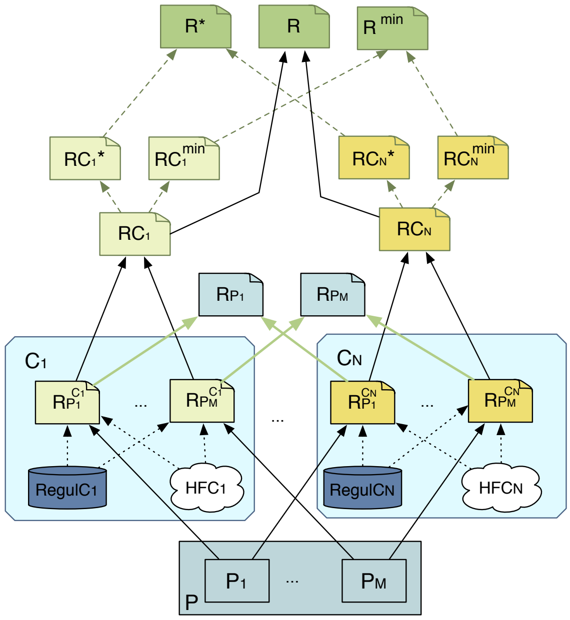

Figure 3: Requirements structuring for the distributed development of products $P_1, \dots, P_M$

Step 0: $RL^{C_j*} = \emptyset$, $X = RL^{C_j}$.

Step 1: If $X \neq \emptyset$, the RL^{C_j*} is complete, otherwise choose a requirement $req \in X$:

- If a copy of *req* already belongs to RL^{C_j*} or r is a weaker version of any requirement from RL^{C_j*}, the set should not be updated on this iteration: $(req \in RL^{C_j*} \lor \exists y \in RL^{C_j*} : req \rightsquigarrow y) \Rightarrow$ RL^{C_j*} *is unchanged*
- If r is a stronger version of any requirement(s) from RL^{C_j*}, we add r to RL^{C_j*} and remove all the weaker requirements: $\exists y_1, \dots, y_K \in RL^{C_j*} : y_1 \rightsquigarrow req \land \dots \land y_K \rightsquigarrow req) \Rightarrow$ $RL^{C_j*} = (RL^{C_j} * \cup req) \setminus \{y_1, \dots, y_k\}$
- If r does not belong to RL^{C_j*} and is neither weaker nor stronger version of any requirement from RL^{C_j*}, we add it to RL^{C_j*} and proceed the procedure with the next requirement from RL^{C_j}: $(req \notin RL^{C_j*} \land$ $\nexists y \in RL^{C_j*} : (req \rightsquigarrow y \lor y \rightsquigarrow req) \Rightarrow$ $RL^{C_j*} = (RL^{C_j*} \cup req)$

Step 2: The *req* element is deleted from the set X: $X = X \setminus req$.

Steps 1 and 2 are then repeated until $X = \emptyset$.

While identifying RL^{min} we will analyse which products' subcomponents can be build once and then reused for the whole product set $P_1, \dots, P_M$. On this basis, we will have more efficient process for the global software and systems development, also having an efficient tracing of requirements changes that might come from the changes in the regulations and laws for countries $C_1, \dots, C_N$. For example, changes in $RegulC_1'$ imply changes in $RLspecific_1$ only, which means that only the country-specific part of the architectural components for C_1 is affected, where any changes in *Regul* might influence the global architecture.

While identifying $RL*$ we will obtain the global view on the the products' requirements, which is not overloaded with the variants of the similar requirements, where some requirements are just weaker versions of other.

5 CONCLUSIONS

This paper introduces our ongoing work on requirements specification and analysis for the geographically distributed software and systems. Developing software and systems within/for different countries or states or even within/for different organisations means that the requirements to them can differ in each particular case, which naturally impacts on the software architecture and on the development process as a whole. Dealing with this diversity and avoiding contradictions and non-compliance, is a very challenging and complicated task. A systematic approach is required. For this reason, we created a formal framework for the analysis of the software requirements diversity, which comes from the differences in the regulations for different countries or organisations. In this paper, we (i) presented our architectural dependency model for the requirements on the distributed development and application, (ii) introduced the core ideas of the corresponding requirements structuring process and requirements analysis in a global context. (iii) discussed the the research and industrial challenges in this field, as well as discussed our solutions and how they are related to the existing approaches.

Future Work: In our future work we will investigate how to extend the presented ideas to the software and systems development that involves hierarchical dependencies between the sets of regulations/laws. This could be the case if see the set $C_1, \dots, C_N$ not only as the set of countries/states, but also as a set of organisations having different internal regulations. Then we have to deal with hierarchical dependencies with many levels, e.g., (1) organisational regulations, (2) state's regulations and laws, (3) country's regulations and laws, where we also need to check which of the regulations are applicable in each particular case.

REFERENCES

Borchers, G. (2003). The software engineering impacts of cultural factors on multi-cultural software development teams. In *International Conference on Software Engineering*, pages 540–545. IEEE Computer Society.

Braun, P., Broy, M., Houdek, F., Kirchmayr, M., Mller, M., Penzenstadler, B., Pohl, K., and Weyer, T. (2014). Guiding requirements engineering for software-intensive embedded systems in the automotive industry. *Computer Science - Research and Development*, 29(1):21–43.

Breaux, T., Anton, A., Boucher, K., and Dorfman, M. (2008). Legal requirements, compliance and practice: An industry case study in accessibility. In *IEEE Conf. on Requirements Engineering*, pages 43–52.

Breaux, T., Vail, M., and Anton, A. (2006). Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In *IEEE International Conf. on Requirements Engineering*, pages 49–58.

Broy, M. and Slotosch, O. (2001). From requirements to validated embedded systems. In Henzinger, T. A. and Kirsch, C. M., editors, *Embedded Software*, volume 2211 of *LNCS*, pages 51–65. Springer.

Broy, M. and Stølen, K. (2001). *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer.

Crnković, I., Schmidt, H., Stafford, J., Heineman, G., and Wallnau, K. (2007). Component-based software engineering of trustworthy embedded systems. *Journal of Systems and Software*, 80(5):641–642.

Egyed, A., Grnbacher, P., and Medvidovic, N. (2001). Refinement and Evolution Issues in Bridging Requirements and Architecture - The CBSP Approach. In *From Software Requirements to Architectures*, pages 42–47.

Ferrari, R. and Madhavji, N. (2007). The impact of requirements knowledge and experience on software architecting: An empirical study. In *The Working IEEE/IFIP Conference on Software Architecture*, pages 16–16.

Glinz, M. (2007). On non-functional requirements. In *IEEE Conf. on Requirements Engineering*, pages 21–26.

Hölzl, F., Spichkova, M., and Trachtenherz, D. (2010). AutoFocus Tool Chain. Technical Report TUM-I1021, TU München.

In, H., Boehm, B., Rodger, T., and Deutsch, M. (2001). Applying winwin to quality requirements: a case study. In *Int. Conf. on Software Engineering*, pages 555–564.

Kiyavitskaya, N., Krausova, A., and Zannone, N. (2008). Why eliciting and managing legal requirements is hard. In *Workshop on Requirements Engineering and Law*, pages 26–30.

Maxwell, J. and Anton, A. (2009). Checking existing requirements for compliance with law using a production rule model. In *Workshop on Requirements Engineering and Law*, pages 1–6.

Nekvi, R., Ferrari, R., Berenbach, B., and Madhavji, N. (2011). Towards a compliance meta-model for system requirements in contractual projects. In *Workshop on Requirements Engineering and Law*, pages 74–77.

Nuseibeh, B. (2001). Weaving together requirements and architectures. *Computer*, 34(3):115–117.

Otto, P. and Anton, A. (2007). Addressing legal requirements in requirements engineering. In *IEEE Conf. on Requirements Engineering*, pages 5–14.

Paech, B., von Knethen, A., Drr, J., Bayer, J., Kerkow, D., Kolb, R., Trendowicz, A., Punter, T., and Dutoit, A. (2003). An experience-based approach for integrating architecture and requirements engineering. In *Workshop on Software Requirements to Architectures*, pages 142–149.

Pohl, K. and Sikora, E. (2007). Structuring the co-design of requirements and architecture. In *Requirements Engineering: Foundation for Software Quality*, pages 48–62. Springer.

Pretschner, A., Broy, M., Kruger, I. H., and Stauner, T. (2007). Software engineering for automotive systems: A roadmap. In *Future of Software Engineering*, pages 55–71. IEEE Computer Society.

Rinke, T. and Weyer, T. (2007). Defining reference models for modelling qualities: How requirements engineering techniques can help. In Sawyer, P., Paech, B., and Heymans, P., editors, *Requirements Engineering: Foundation for Software Quality*, pages 335–340. Springer.

Siena, A., Perini, A., Susi, A., and Mylopoulos, J. (2009a). A meta-model for modelling law-compliant requirements. In *Workshop on Requirements Engineering and Law*, pages 45–51.

Siena, A., Perini, A., Susi, A., and Mylopoulos, J. (2009b). Towards a framework for law-compliant software requirements. In *International Conference on Software Engineering. Companion Volume*, pages 251–254.

Spichkova, M. (2008). Refinement-based verification of interactive real-time systems. *ENTCS*, 214(0):131 – 157. BAC-FACS Refinement Workshop.

Spichkova, M. (2011). Architecture: Requirements + Decomposition + Refinement. *Softwaretechnik-Trends*, 31(4).

Spichkova, M. (2012). Human Factors of Formal Methods. In *IADIS Interfaces and Human Computer Interaction (IHCI 2012)*, pages 307–310. IADIS Press.

Spichkova, M., Hölzl, F., and Trachtenherz, D. (2012). Verified system development with the autofocus tool chain. In *Workshop on Formal Methods in the Development of Software*, EPTCS, pages 17–24.

Spichkova, M., Schmidt, H., and Peake, I. (2013). From abstract modelling to remote cyber-physical integration/interoperability testing. In *Improving Systems and Software Engineering Conference*.

van Lamsweerde, A. (2008). Requirements engineering: From craft to discipline. In *International Symposium on Foundations of Software Engineering*, SIGSOFT '08/FSE-16, pages 238–249. ACM.

Yin, Q., Madhavji, N., and Pattani, M. (2013). Eros: an approach for ensuring regulatory compliance of process outcomes. In *Workshop on Requirements Engineering and Law*, pages 21–24.